Populate your Game Scene!

Julien Pettré

Populate Your Game Scene

Julien Pettré

INRIA Rennes - Bretagne Atlantique
Campus de Beaulieu, 35042 Rennes, France
julien.pettre@inria.fr

Abstract. We describe a method for populating large virtual environments with crowds of virtual pedestrians. Pedestrians are distributed in the environment by giving them goal destinations to reach. In order to facilitate this population setup, we assign identical destinations to batch of pedestrians, resulting in navigation flows. In order to preserve individuality of pedestrians, navigation is planned with variety and each pedestrians follows its own unique trajectory. This specific navigation planning technique answers user's queries very rapidly. As a result, impact of the insertion of a new navigation flow in the environment is directly observable, and interactive design is achieved. Once the problem of design solved, we address the problem of simulating the virtual population navigation. A level-of-details strategy is used to achieve on-line real-time simulation of large crowds, formed by up to tens of thousands of pedestrians. We illustrate our method on the typical example of a virtual city, and discuss our approach.

1 Introduction

1.1 Motivations

Many video games action take place in large virtual environments such as buildings, districts, entire cities or outer landscape. Players and virtual opponents are present in these environments, however, if the environment remains only populated with players, real or virtual, the scene may still look empty. Scene liveliness can be improved by filling up the place with a population of virtual inhabitants. A large number of them may be required to achieve this goal, even more when scenes are large. As a result, some aspects of this problem are close to the one of crowd simulation. However, video games applications have specific needs: among them, the need for a *eye-believable population simulation* (that can be opposed to the need for realistic results, and that also eliminate some classes of crowd simulation techniques such as cell-automata); the need for *interactivity* between the player and the population, which induces the need for on-line simulation; and finally, the need for giving *control* on the population activity to game designers.

Our objective is to provide tools for creating and animating large virtual populations, in order to make large environments alive. Some questions need addressing: how to setup a crowd and get control on its activity? how to simulate efficiently a crowd on the performances point of view? In the next section, we

give a very brief overview of existing techniques for crowd design, visualization, simulation and motion planning. However, in this article, we will specifically focus on populations of pedestrians with behavior restricted to navigation. We will give key-ideas for designing large navigating crowds from simple and few high-level directives, and describe a crowd simulator exploiting a level-of-details strategy in order to reach high performances.

1.2 Related Work

Crowd Visualization and Design. Crowd Visualization is a specific problem with various solutions [1], [2], [3]. Briefly, these solutions rely on two major concepts. The first one is the level-of-details concept which consists in representing entities on the screen with a varying complexity, depending on its visibility and distance to the spectator's point of view; the more the entity is viewable (close and centered in the screen), the more its representation is accurate, and inversely. The second one is the use of impostors. Impostors are 2D images of entities which are used in place of 3D models when displaying the entity on the screen, allowing important computation time saves. It is possible to use such simple representations while preserving variety for visual aspect of individuals. CrowdBrush is a user-friendly tool for Crowd Design [4]. This tool is mainly dedicated to give visual aspects to many individuals in an easy manner, but also allows the distribution in space of the crowd and the definition of the inhabitants activity. Using CrowdBrush, goal destinations can be assigned to individuals, however, the user is in charge of providing a collision free path. Compared to this useful tool, we provide a solution to compute automatically collision-free paths for batches of pedestrians.

Crowd Simulation. Simulating crowds first consists in computing trajectories for several entities while taking into account their interactions and presence of obstacles in the environment. We briefly present 3 general methods to address crowd simulation among others: rule-based systems, particle systems and cell-automaton methods. Reynolds' rule-based model is a popular approach, allowing to compute motions for coherent groups [5]. Each entity's motion is influenced by other entities in its vicinity resulting in a global motion that can be compared to flock of birds or fishes. Reynolds declined later his approach in [6] to propose a complete set of steering behaviors such as seeking, fleeing, pursuing, etc. Using Particles Systems is another popular manner to simulate crowds [7], interactions between particles may be modeled by forces such as in [8], or by using potential fields [9]. Finally cell-automaton techniques relies on a floor-field structure, a regular discretization of the navigable space looking like a grid. Each cell of this floor field may be occupied or free. Each time-step, the content of occupied cells has a probability to be transferred toward a neighbor cell, modeling people's motion.

Crowd Control. We intend to populate environments with crowds of pedestrians with goal-directed navigation; this problem is thus reduced to a navigation planning problem. An exhaustive description of existing motion planning algorithms

can be found in [10]. Briefly, three major classes of solutions can be distinguished. Firsts are *reactive methods*: the key-principle of these techniques is to model goals as attractive and obstacles as repulsive potentials [11]; the moving entity follows the gradient of the resulting potential field in order to reach its goal. This kind of techniques suffers some local minimum deadlocks, however, they are simple to implement and practical to use. *Sampling-based* motion planning is the second major class of approaches. These methods explore partially the navigable space in order to capture interconnected collision-free positions and paths into a graph structure (commonly called a roadmap). The well-known PRM (Probabilistic Roadmaps Methods, [12]) and RRT (Rapidly-Exploring Random Tree [13]) methods explore the free space in a random manner. A the opposite, deterministic approaches are based on a complete exploration of the free space, however, they can use discrete representation or complete representation of this space, resulting respectively in partial or complete solutions. Deterministic approaches result in a data structure, generally a graph, capturing both the geometry and the topology of the free environment. Navigation planning queries are solved using graph search algorithms. Deterministic approaches were recently used to adress the crowd navigation problem in [14].

1.3 Overview

Our approach allows a designer to populate a scene by adding progressively navigation flows in the environment. Navigation flows are subsets of pedestrians navigating back and forth endlessly between two given locations. The designer controls the distribution of the population in the environment only by selecting these destinations, and the pedestrians groups size (i.e., the number of individuals composing the group). By answering very rapidly designers' queries, they can immediately see the impact of the insertion of a new navigation flow, and compose a population by trial and error. There is no theoretical limitation on the number of flows, as the simulation complexity is mainly dependent on the total number of pedestrians composing the whole population.

These objectives are reached by combining 3 major ingredients:

1. *Navigation Graphs*: they result from a specific environment analysis which is similar to a cell-decomposition technique. Navigation Graphs are computed at a preprocessing stage, and support future navigation planning queries.
2. *Navigation Flows*: as defined above, they are sets of navigating pedestrians with common destinations. However, we want to preserve individuality in trajectories followed by each pedestrian. For this reason, we developed a specific navigation planning technique which computes a variety of solution paths. Doing so, from a unique navigation planning query, we can generate a variety of solution paths.
3. *Crowd Simulator*: it exploits a Level-of-Detail strategy, which allows to dispatch the available computation resources at the best. As for crowd visualization techniques, the key-idea is to get high-quality simulation in front of the user's point of view whereas the quality is progressively lowered in far and invisible areas.

Next sections detail each of these three ingredients. We demonstrate the efficiency of our approach in the Results section.

2 Navigation Graphs

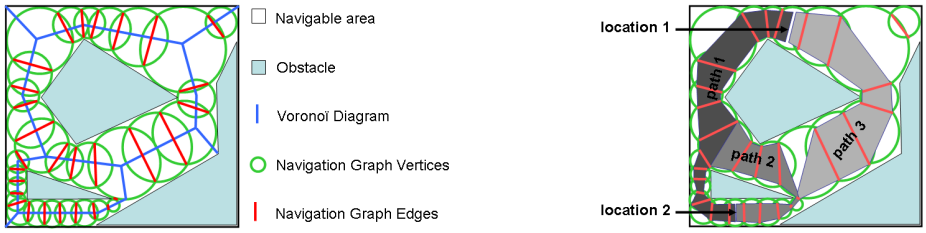

Fig. 1. *Left Image:* 2-D representation of a Navigation Graph. *Right Image:* Result of a Navigation Flow query: 3 Navigation Paths are found, each being a sequence of gates delimiting a corridor joining the 2 locations given as input of the query

Navigation Graphs ($\mathcal{NG}$) is a graph structure capturing the topology and the geometry of the *navigable space* of a given environment. Navigable space is the flat part of an environment (flat enough to allow human walk, according to a user-defined threshold slope) free of any obstacle.

A $\mathcal{NG}$ vertex captures a portion of the navigable space with a vertical cylindrical shape, high enough to include a human (typical height is 2 meters). A $\mathcal{NG}$ edge models a connection between two navigable cylinders, meaning they have a common portion of navigable space. Geometrically, passages between navigable cylinders are rectangular gates (with size as great as possible) included in the intersection of the two connected cylinders.

An example of Navigation Graph is displayed in Fig. 1 for a simple 2-D environment: obstacles are the blue regions of the figure, while the navigable area is left blank. The Voronoï diagram of the navigable area is displayed as well. $\mathcal{NG}$ vertices are represented as green circles centered on the Voronoï diagram. Edges are represented as red gates between overlapping vertices.

$\mathcal{NG}$ are computed automatically from the environment geometry and few user-defined parameters, as described in [15], in 5 successive steps:

1. *environment geometry sampling*: we create a regular grid of points matching the environment surfaces. As multi-layered environments may be considered, a simple elevation is insufficient since several elevations may correspond to given horizontal coordinates.
2. *grid mesh*: two neighboring grid points are interconnected when the slope of the in-between space is beneath the user-defined maximum slope angle and free of obstacle. With this stage, we provide a mesh capturing the navigable space in a discrete manner.

3. *clearance map*: we compute the clearance for each grid point, i.e. the distance to the nearest obstacle or high-slope. Given the the grid mesh computation method, this distance is approximated by the distance to the nearest grid point not connected to all of its direct neighbors. Indeed, The lack of connection reveals the presence of an obstacle or a high-slope.
4. $\mathcal{NG}$ *deduction*: we then use a subset of grid points to compute the $\mathcal{NG}$ vertices. A graph vertex (cylinder) is created from a grid point by using it as centre and its corresponding clearance as radius. The grid point with maximum clearance is selected to create the graph vertex and all the grid points covered by the vertex are disabled for selection. The process is then reiterated until no more grid points remain for selection. Doing so, a majority of cylinders are centered on the Voronoï diagram corresponding to the environment.
5. *visibility pre-computation*: for each $\mathcal{NG}$ vertex, we compute the visibility of all other vertices according to the four main cardinal points. We can then use this information for visibility queries between moving entities.

On one hand, the use of an intermediate grid during computation makes the solution incomplete: navigable space is partially captured. But on the other hand, this discretization makes our method robust to complex and various environments.

3 Navigation Flows

A designer populates a virtual scene by creating as many *Navigation Flows* as desired. A Navigation Flow is a subset of pedestrians navigating endlessly between two locations of the environment. By choosing the number of pedestrians composing the flow as well as the locations to be joined, the designer is able to control the dispatching pedestrians in the scene. For example, the distribution of pedestrians into a virtual city is done according to the importance of some specific centers of interest, such as public buildings in a virtual city, shops, attractions, etc.

Collision free paths joining two locations can be computed easily using navigation graphs by simple graph search algorithms, such as A* or Dijkstra's. The search results in a *Navigation Path*. A solution path is composed by a sequence of connected edges to follow, i.e., a sequence of rectangular passages to cross. This sequence of passages delimits a solution corridor joining the two desired locations. The width of this corridor allow to: dispatch people, individualize their trajectories and let them avoid each other.

The individualization of navigation trajectories is important in order to preserve the believability of the obtained crowd motion. Indeed, if all pedestrians follow exactly the same path when navigating, strange queue formations appear as well as jams at some places in the scene with a negative impact on the resulting crowd motion. Following this idea, a single Navigation Path is even insufficient to drive a navigation flow. For that, we search for alternative paths that will

finally compose Navigation Flows: a second level of variety in trajectories is thus obtained.

This search for alternative paths is illustrated in Fig. 1. A navigation flow is created between locations 1 and 2 as indicated on the right image of the figure. A first navigation path (the shortest) is found. As mentioned before, this navigation path is geometrically similar to a corridor joining the two locations. But, still in Fig. 1, one can see that paths 2 and 3 also compose the navigation flow. These two alternative paths will allow pedestrians to join their destinations passing by one side or another of obstacles in the environment.

The existence of several navigation paths in a navigation flow is useful when jams appear during the simulation. The way pedestrians - belonging to an identical navigation flow - are dispatched inside a given solution path, and the way they are dispatched among the alternative navigation path is detailed in next Section. An extensive description of the Navigation Flows computation technique is given in [16].

4 Scalable Pedestrian Simulation

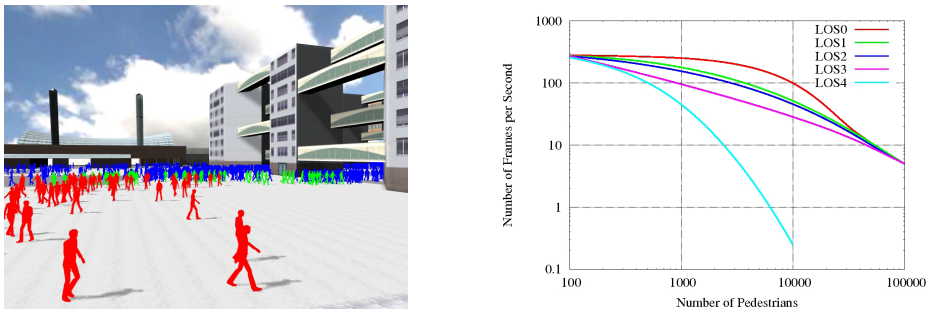

Fig. 2. *Left Image:* Distribution of different levels-of-details in scene. Most visible pedestrians (red) avoid each other, whereas far pedestrians (green and blue) do not avoid themselves and are updated at lower rates. *Right plot:* Performances obtained for each Level of Simulation LoS according to the number of pedestrians. LoS4, the most accurate, has quadratic complexity due to collision avoidance between pedestrians.

Once a navigation flow is created, simulation starts. During simulation, pedestrians' position is updated according to the path they are following and their position relatively to spectator's point of view. In this Section we describe:

1. How an individual trajectory is computed from the characteristics of the flow they belong to?
2. How the position-update accuracy is tuned according to the centrality and visibility of the corresponding pedestrian into the screen.

From Navigation Flow to Navigation Path A pedestrian belonging to a navigation flow first chooses a navigation path among the available ones (i.e., one of the corridor as displayed in Fig. 1). This choice is made each time the pedestrian is at one of the extremity of the flow and have to go back (i.e., when at location 1 going to location 2 or inversely in Fig. 1). This choice is irrevocable until he reaches his destination. Choice is based on the estimated travel time for each path belonging to the flow. Estimation takes into account the total length and density of population along the path, as described in [17]. Using a velocity /density statistical law $v = f(density)$, we can estimate the required time for crossing each area. By summing these crossing times, we compute the estimated travel time for each path. For example, a path composed by n edges has a travel time estimated to: $ETT_{path} = \sum_{i=0}^{n} ETT_{edge_i}$ with $ETT_{edge_i} = \frac{length_i}{v=f(density)}$. The local densities of population are not varying rapidly, as a result, the estimated travel times are updated at low frequencies. Finally, pedestrians choose the path currently having the lowest estimated travel time.

From Navigation Path to Individual Path The distribution of pedestrians following a same navigation path is simply done according to an individual parameter p. This individual parameters (ranging from 0 to 1) determines a tendency of the pedestrian to walk on the left or the right of the followed corridor. More precisely, pedestrians follow way-points placed at each gate to cross. The position of the way-point WP within the gate AB is done as follow: $WP = A + p\boldsymbol{AB}$.

Levels of Simulation Main objective of the simulation is to get believable results with high performances. High performances can be obtained by applying a strategy of levels-of-detail to the simulation. The principle is to focus the dedicated computation resources where it is the most needed, i.e., in the most visible parts of the environment. In areas at a far distance of the user point-of-view, or in the invisible parts of the environment, the simulation can be progressively simplified in order to save precious computation-time. Two major components in simulations allow to diminish the complexity of the simulation:

1. *update-rates:* Updating each individual at each simulation step results in a complexity which is linear with the number of simulated pedestrians. However, the spectator is able to see only a few part of pedestrians at a given instant with respect to his field of vision and the presence of obstacles occluding parts of the environment. We first improve the simulation performances by updating at high-frequencies positions of the most visible pedestrians while the other ones benefit from less frequent updates.
2. *reactive collision avoidance:* reactive collision avoidance between pedestrians moving in the same area is probably the most time consuming task in the simulation, with quadratic complexity. Collisions between pedestrians is required where the attention of the spectator is focused, however, elsewhere, collisions are not checked. Note that pedestrians are already dispatched in the environment using our specific Navigation Flow technique (both on several paths and inside each navigation path). As a result, at far distances (and

obviously in invisible areas), the lack of collision-checks between pedestrians is not easily detectable.

Figure 2 illustrates our Level-of-Simulation (LOS) strategy. On the left image, the distribution of LOS is displayed: most visible pedestrians are in red. For them, collision avoidance is enabled and their situation is updated at each simulation step. Obtained performances are plotted on the right (LOS4 plot). For pedestrians who are far from the user point of view, the collision avoidance is first disabled (pedestrians in green on the left image, corresponding to LOS3 on the plot). Then, situation update rate is progressively lowered (LOS2, 1 and 0, pedestrians in blue in the image). In conclusion, Level-of-Simulations is a key principle in order to focus available computation resources where they are the most needed, and to improve drastically simulation performances: with low update rates and no collision avoidance, 10 000 pedestrians can be simulated at 100Hz.

5 Example and Discussion

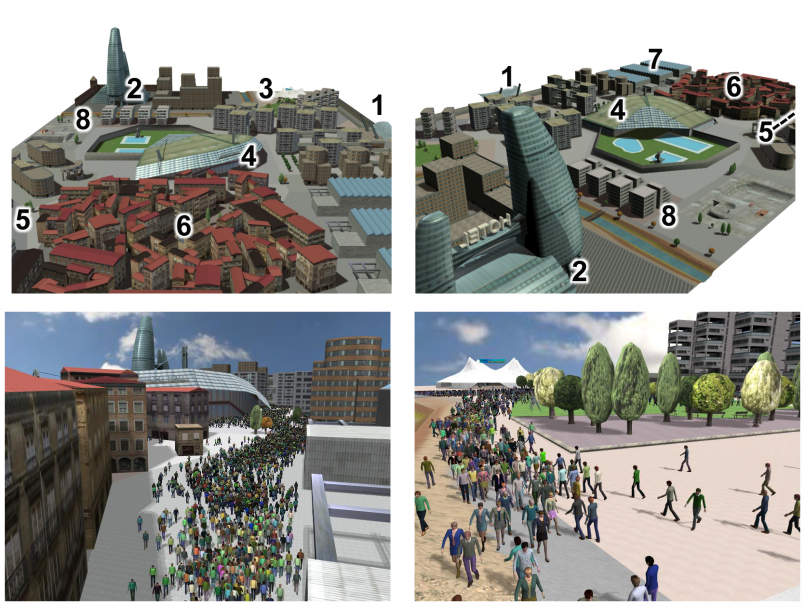

Fig. 3. *Top images:* Views of a virtual city, with 8 centers of interests corresponding to most important buildings (hotel, train station, church, circus, etc.). *Bottom images:* Views of the virtual population generated by creating several navigation flows between the centers of interest.

In order to illustrate our method, we consider the following example of a virtual city displayed in Figure 3. This environment is large, 640×420 meters, and composed by approximately 100'000 triangles. A Navigation Graph is computed for this scene in 20 minutes approximately, composed by 5017 vertices and 8003 edges. Four visual output of the crowd simulation, we use the YAQ crowd visualization system developed at EPFL-VRlab.

In order to populate this environment, 7 navigations flows are sufficient. They join some main centers of interest into the city, such as an hotel, a train station, a circus, the old center of the city, etc. Each of the 7 Navigation Flow is composed by 5000 pedestrians, resulting in a virtual population made of 35'000 pedestrians. During Population Setup, each Navigation Flow is computed in less than a second, and as the simulation and visualization runs during this stage, we can evaluate immediately the impact of the insertion of new flows. Despite the small number of flows to populate this scene, all the places of this environment are occupied with pedestrians. This results from our specific dispatch method in Navigation Flow computations.

6 Conclusion

We presented a Navigation Graph based approach for populating scenes with crowds of pedestrians. The principle is to populate semi-automatically environments by planning the navigation of large crowds of pedestrians. Few high-level directives allow to populate large scenes with numerous pedestrians. We proposed efficient techniques in order to allow: first, interactive design of the population by answering quickly to user's navigation queries, and second, on-line simulation with real-time rates.

For efficient Navigation Planning, the key-idea is to search for varieties of solution. This allow to derive solution paths in order to individualize pedestrians' trajectories with no supplementary computation cost.

For efficient simulation of pedestrians, the key-idea is to use a level-of-details strategy in order to maintain a high simulation quality where it is the most needed, i.e., in front of the spectator, whereas the simulation quality is lowered elsewhere.

Our method fits the video-games field, and is applicable to the problem of populating large game scenes with large virtual population of pedestrians with low computation costs and believable results.

Acknowledgments

This work was realized in 2005-06 at EPFL-VRlab, directed by Pr. Daniel Thalmann. The project was funded by the European Research Community, by a Marie-Curie Action EIF 2005-mobility. The author would like to thank B. Yersin and J. Maïm who developed the crowd visualization system used in our examples, and M. Clavien and R. Krummenacher for environment design.

References

1. Aubel, A., Boulic, R., Thalmann, D.: Real-time display of virtual humans: Levels of details and impostors. IEEE Transactions on Circuits and Systems for Video Technology **10**(2) (2000) 207–217
2. Tecchia, F., Loscos, C., Chrysanthou, Y.: Visualizing crowds in real-time. Computer Graphics Forum **21**(4) (December 2002) 753–765
3. Dobbyn, S., Hamill, J., O'Conor, K., O'Sullivan, C.: Geopostors: A real-time geometry/impostor crowd rendering system. In: I3D'05: Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. (2005) 95–102
4. Ulicny, B., de Heras, P., Thalmann, D.: Crowdbrush: Interactive authoring of real-time crowd scenes. SCA'04: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2004) 243–252
5. Reynolds, C.W.: Flocks, herds, and schools: A distributed behavioral model. In Stone, M.C., ed.: Computer Graphics (SIGGRAPH '87 Proceedings). Volume 21. (July 1987) 25–34
6. Reynolds, C.W.: Steering behaviors for autonomous characters. In: The proceedings of the 1999 Game Developers Conference. (1999)
7. Bouvier, E., Guilloteau, P.: Crowd simulation in immersive space management. In: Proceedings of the Eurographics workshop on Virtual environments and scientific visualization '96, London, UK, Springer-Verlag (1996) 104–110
8. Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Phys. Rev. E **51**(5) (May 1995) 4282–4286
9. Treuille, A., Cooper, S., Popovic, Z.: Continuum crowds. In: Proceedings of ACM SIGGRAPH. (2006)
10. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge, U.K. (2006) Available at http://planning.cs.uiuc.edu/.
11. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. International Journal of Robotics Research **5**(1) (1986) 90–98
12. Kavraki, L., Svestka, P., Latombe, J.C., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. Robotics and Automation, IEEE Transactions on **12**(4) (Aug. 1996) 566–580
13. Kuffner, J.J., J., LaValle, S.: Rrt-connect: An efficient approach to single-query path planning. In: Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on. Volume 2. (24-28 April 2000) 995–1001
14. Sud, A., Andersen, E., Curtis, S., Lin, M., Manocha, D.: Real-time path planning in dynamic virtual environments using multiagent navigation graphs. Visualization and Computer Graphics, IEEE Transactions on **14** (2008) 526–538
15. Pettré, J., Laumond, J.P., Thalmann, D.: A navigation graph for real-time crowd animation on multilayered and uneven terrain. In: First International Workshop on Crowd Simulation (V-CROWDS05). (2005)
16. Pettré, J., Grillon, H., Thalmann, D.: Crowds of moving objects: Navigation planning and simulation. In: Robotics and Automation, 2007 IEEE International Conference on. (10-14 April 2007) 3062–3067
17. Pettré, J., de Heras Ciechomski, P., Maïm, J., Yersin, B., Laumond, J.P., Thalmann, D.: Real-time navigating crowds: scalable simulation and rendering: Research articles. Comput. Animat. Virtual Worlds **17**(3-4) (2006) 445–455